CITS3003 graphics and animation project report

22289211 Haoran zhang
22289267 Hongfeng Wang

Functionality

The program work well when we test it. I did A-E individually in part1 and the leftover part1 were finished by my partner Hongfeng.
The model animation and movement features are fully functional and appear to work without any problems. I did A and D in part 2. My partner did B and C in part 2.

implementation

project1:

A. Following the question, I know that changing the display () function can fix the problem. Looking around the function and find out this comment.

```
// Set the view matrix.  To start with this just moves the camera
// backwards.  You'll need to add appropriate rotations.
//A
```

In a result, I know I should do the rotation in the next step. According to the lab and the hint in the problem A, I change the variable 'view' like this.

```
//A
view = Translate(0.0, 0.0, -viewDist) * RotateX(camRotUpAndOverDeg) *
    RotateY(camRotSidewaysDeg);
```

B. It is similar to the step A. The comment and the instruction give me the idea.

```
// Set the model matrix - this should combine translation, rotation and scaling based on
    what's
// in the sceneObj structure (see near the top of the program).
//B
mat4 rotation=RotateX(-sceneObj.angles[0]) * RotateY(-sceneObj.angles[1]) *
    RotateZ(-sceneObj.angles[2]);
mat4 model = Translate(sceneObj.loc) * rotation * Scale(sceneObj.scale);
```

C. According to the information in the function materialMenu and makeMenuM, I imitated it and do the functions named adjustAmbientDiffuse and adjustSpecularShine.

```
542 static void adjustAmbientDiffuse(vec2 a) {
543     sceneObjs[toolObj].ambient = max(0.0f, sceneObjs[toolObj].ambient + a[0]);
544     sceneObjs[toolObj].diffuse = max(0.0f, sceneObjs[toolObj].diffuse + a[1]);
545 }
546
547 static void adjustSpecularShine(vec2 b) {
548     sceneObjs[toolObj].specular = max(0.0f, sceneObjs[toolObj].specular + b[0]);
549     sceneObjs[toolObj].shine = max(0.0f, sceneObjs[toolObj].shine + b[1]);
550 }
```

D, E. we redefine the clipping volume as follows: the bottom (-nearDist) and top (nearDist) clipping planes fit with the height of the new window; the left and right clipping planes become - nearDist *width/height and nearDist *width/height. Thus, when width > height, there are extra spaces on the left and right edges of the window not being used. And also, the lab has the same content.

```
GLfloat nearDist = 0.02;
if(width < height){
    projection = Frustum(-nearDist,
                         nearDist,
                         -nearDist*(float)height/(float)width,
                         nearDist*(float)height/(float)width,
                         nearDist, 500);
}
else{
    projection = Frustum(-nearDist*(float)width/(float)height,
                         nearDist*(float)width/(float)height,
                         -nearDist,
                         nearDist,
                         nearDist, 500);
}
```

F. In order to apply light source reducing linearly as its distance increase, I do this.

```
//F
float distances = length(Lvec1)*0.4; //sqrt(length(Lvec1))/15.0 + 1.0 ;
//G
//vec4 color;
//no mutiply the distance only division can make the light1 light
//the distances is more larger, the light of light1  is more weak
vec4 color = vec4((globalAmbient + (ambient1 + diffuse1)/distances + (ambient2 +
    diffuse2)),1.0);
//color.a = 1.0;
```

G. Following the instruction, I move the lighting calculation from vshader to fshader. In this process, I left the parameters which use attribute variable in vshader, and modified the declaration of the variables in the top.
vshader

```
attribute vec3 vPosition;
attribute vec3 vNormal;
attribute vec2 vTexCoord;

varying vec2 texCoord;
//varying vec4 color;
//varying vec3 Lvec;
varying vec3 pos;
varying vec3 N;

//uniform vec3 AmbientProduct, DiffuseProduct, SpecularProduct;
uniform mat4 ModelView;
uniform mat4 Projection;
uniform vec4 LightPosition;
uniform float Shininess;
```

fshader

```
//varying vec4 color;
varying vec2 texCoord;  // The third coordinate is always 0.0 and is discarded

//varying vec3 Lvec;
varying vec3 N;
varying vec3 pos;

uniform sampler2D texture;

uniform vec3 AmbientProduct, DiffuseProduct, SpecularProduct;
uniform mat4 ModelView;
//uniform mat4 Projection;
uniform vec4 LightPosition;
uniform float Shininess;
```

H. In order to make the highlights tending towards white rather than the colour of the object, we need to separate the specular and the colour in color. It means that we need to change the way of them to pass into the shader. Applying the specular to the fcolor because we need to avoid changing the color of object's texture.

I. following the instruction, using the sceneOjb[2] as the second light, and going through the scene-start document to add the light 2 imitating the light 1. However, we should take care of the this,

```
SceneObject lightObj1 = sceneObjs[1];
vec4 lightPosition1 = view * lightObj1.loc ;

glUniform4fv( glGetUniformLocation(shaderProgram, "LightPosition1"),
              1, lightPosition1);
CheckError();
//stepI
SceneObject lightObj2 = sceneObjs[2];
vec4 lightPosition2 = RotateX(camRotUpAndOverDeg) * RotateY(camRotSidewaysDeg) *
    lightObj2.loc ;

glUniform4fv( glGetUniformLocation(shaderProgram, "LightPosition2"),
             1, lightPosition2);
CheckError();
```

The second light should be directional. So only the rotation needs to implement.

Project2:

A.I have done this in part 1

```
    //stepB texture scale
    gl_FragColor = color * texture2D(texture,texCoord*texScale)+(specular1/distances) +
        specular2;;
}
```

B. Following the instruction in the website, I create three different human model by the software called MakeHuman.

C. Following the instruction in the website, I create three different animation by the software called Blender.

D. firstly, we need to give the speed to the animation in the function called addObject().

```
    //part2
    //if the object is animated,initialize the speed of the animation
    if (id >= 56) {
        sceneObjs[nObjects].walkSpeed =20.0;
    } else {
        sceneObjs[nObjects].walkSpeed = 0.0;
    }
```

And then modifying the drawMesh()

```
393     loadMeshIfNotAlreadyLoaded(sceneObj.meshId);
394
395     aiMesh *mesh = meshes[sceneObj.meshId];
396     const aiScene *scene = scenes[sceneObj.meshId];
397
398     float poseTime = 0.0;
399     float animationDuration;
400
401     if (sceneObj.meshId >= 56) {
402         //get the miliseconds
403         float elapsedTime = glutGet(GLUT_ELAPSED_TIME)*0.001;
404         //get the animation duration
405         if(mesh->mNumBones == 0){
406             animationDuration=0.0;
407         }
408         aiAnimation *animation = scene->mAnimations[0];
409         animationDuration = animation->mDuration;
410         //------------------------
411         float animationTime = fmod(elapsedTime * sceneObj.walkSpeed,animationDuration);
412         poseTime = fmod(animationTime,animationDuration);
413     }
```

After adding the codes in the instruction, we learn that the calculateAnimPose() calculates the bone transformations for a mesh at a particular time in an animation (in scene). We need to give the parameter “poseTime” to calculateAnimPose(). All we need to do is to calculate each ‘pose’ taking how long time.

Each aiMesh object contains an animation array and an animation contains the attribute call mDuration. These can be used to determine the length of the animation in miliseconds.

Next, making the animation bigger.

```
//part 2 change the scale of the animation
if (sceneObj.meshId >= 56) {
    model = Translate(sceneObj.loc) * rotation * Scale(sceneObj.scale*10);
}
else{
    model = Translate(sceneObj.loc) * rotation * Scale(sceneObj.scale);
}
```

Finally, we add a new menu called "Speed of animation", and its functionality is changing the speed of the animation by using mouse.

```
695 static void adjustWalkSpeed(vec2 walk){
696     sceneObjs[currObject].walkSpeed += walk[0];
697 }
714     if (id == 30 ){
715         setToolCallbacks(adjustWalkSpeed, mat2(20, 0, 0, 15),
716                          adjustWalkSpeed, mat2(20, 0, 0, 15));
717     }
```

The change of the gnatidread.h

I change the gnatidread.h in way showing below the picture

```
//added the condition to make the scene moving from the last move
static void doToolUpdateXY(int x, int y) {
    if (currButton == GLUT_LEFT_BUTTON || currButton == GLUT_MIDDLE_BUTTON) {
        vec2 currPos = vec2(currMouseXYscreen(x,y));

        if (currButton==GLUT_LEFT_BUTTON && prevPos.x != 0 && prevPos.y != 0)
            leftCallback(leftTrans * (currPos - prevPos));

        else if (prevPos.x != 0 && prevPos.y != 0)
            middCallback(middTrans * (currPos - prevPos));

        prevPos = currPos;
        glutPostRedisplay();
    }
}
```

Adding 2 more condition to make the scene move from the last location rather than the original location.

Reflection

When we initially start the part 1, we feel that it is kind of hard to us to implement the requitements in the website. Once we go through the whole the scene-start.cpp and having an ambiguous concept with this program, the thing becomes easier than before. The hardest part in this program is numerical value in some function. We needed to change it a lot of time to find out a appropriate value. Actually, when we finished the first part of the project, the project looks more transparent to us. And we have a deeper understanding to the code. In a result, according to the hint in the comment of the codes, we finished it faster than part 1.